

SALUS SECURITY

DEC 2024



CODE SECURITY ASSESSMENT

ZEROBASE

Overview

Project Summary

- Name: ZeroBase - Vault
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
 - <https://github.com/ZeroBase-Pro/zerobase-vault>
- Audit Range: See [Appendix - 1](#)

Project Dashboard

Application Summary

Name	ZeroBase - Vault
Version	v1
Type	Solidity
Dates	Dec 16 2024
Logs	Dec 16 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	0
Total informational issues	1
Total	2

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction	4
1.1 About SALUS	4
1.2 Audit Breakdown	4
1.3 Disclaimer	4
Findings	5
2.1 Summary of Findings	5
2.2 Notable Findings	6
1. Centralization risk	6
2.3 Informational Findings	7
2. Use of floating pragma	7
Appendix	8
Appendix 1 - Files in Scope	8

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (<https://t.me/salusec>), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Centralization risk	Medium	Centralization	Pending
2	Use of floating pragma	Informational	Configuration	Pending

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Centralization risk	
Severity: Medium	Category: Centralization
Target: <ul style="list-style-type: none">- src/Vault.sol	

Description

The contract has multiple centralized logic:

- The emergencyWithdraw() function allows the admin to withdraw all tokens from the contract.
- When users claim rewards, a pending request is generated, and the project team must manually replenish tokens in the Vault to fulfill these claims.

If the admin's private key is compromised, an attacker could drain all tokens from the Vault. Additionally, relying on plain EOA accounts for privileged roles exacerbates the risk, raising concerns about the security of user funds.

Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

2.3 Informational Findings

2. Use of floating pragma

Severity: Informational

Category: Configuration

Target:

- src/Vault.sol
- src/Utils.sol

Description

```
pragma solidity ^0.8.28;
```

The `Vault` and `Utils` contract use a floating compiler version `^0.8.28`.

Using a floating pragma `^0.8.28` statement is discouraged, as code may compile to different bytecodes with different compiler versions. Use a locked pragma statement to get a deterministic bytecode. Also use the latest Solidity version to get all the compiler features, bug fixes and optimizations.

Recommendation

It is recommended to use a locked Solidity version throughout the project. It is also recommended to use the most stable and up-to-date version.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit [acc403f](#):

File	SHA-1 hash
utils.sol	495320c4bd0a45e709f5e59cdfd30f97cfdb07d3
Vault.sol	54e945b87663f1cb490b1aff83e3a57bd2cab548